UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/649,270	08/28/2000	Lawrence A. Crowl	SUN1P380/P4501	6759

22434 7590 09/13/2004

BEYER WEAVER & THOMAS LLP
P.O. BOX 778
BERKELEY, CA 94704-0778

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2124	

DATE MAILED: 09/13/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 10/03)

Office Action Summary	Application No. 09/649,270	Applicant(s) CROWL ET AL.
	Examiner Tuan A Vu	Art Unit 2124

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *01 June 2004*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,4-10,12-16 and 19-22* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,4-10,12-16 and 19-22* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)	4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)	5) ☐ Notice of Informal Patent Application (PTO-152)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.	6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 6/01/2004.

As indicated in Applicant's response, claims 1, 4-6, 10, 12-13, 19-21 have been amended, claims 3, 11, 18 canceled, and claim 22 added. Claims 1, 4-10, 12-16, 19-22 are pending in the office action.

Claim Rejections - 35 USC § 103

2. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

3. Claims 1, 4-6, 10, 12-14, 16, 19, 21 and 22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Unger et al, USPN: 5,991,173 (hereinafter Unger), in view of Ainon, "Storing text using integer codes", 1986, *Proceedings of the 11th coference on Computational linguistics* (hereinafter Ainon); and further in view of Storer et al., "The Macro Model for Data Compression", 1978, *Proceedings of the 10th Annual ACM symposium on Theory of computing* (hereinafter Storer).

As per claim 1, Unger discloses a method of generating compiler products (e.g. Fig. 7) in a compressed form, said method comprising:

receiving a source program including one or more program symbols and non-program information (e.g. *vocabulary word, token* – col. 9, lines 5-30; *numeric string, currency symbols* – col. 10, lines 23-39; steps *212-214, 222* – Fig. 8 – Note: non-program symbols are pointer information for tag correspondence);

encoding a program symbol name to produce an encoded program name (see Fig. 8; Note: some form of encoding of textual and numerical symbols is inferred from encoding text strings of HTTP page/document under HTML format and protocol - e.g. col. 4, lines 25-55 – and the fact of compressing symbol name or text of HTML source file implicitly discloses a form of encoding using a algorithm) , without changing the non-program symbol information (e.g. step 222 – Fig. 8);

producing a compressed compiler product based on at least the compressed compiler related information (e.g. step *218, 219*, Fig. 8; col. 12, lines 1-6).

But Unger does not disclose generating a differential name for the encoded program symbol relative to a base symbol for the program symbol, the differential name having a reduced-size format as compared to the encoded program symbol name; and producing a compressed compiler product including the generated differential name.

Nor does Unger disclose that producing the compiler product includes the generated differential name.

Compressing technique by averting redundancy in identifying common or repeated sequences of atomic elements or characters within a stream of data or a sequence of tokens, i.e. characters of natural language or encoded strings, was a known concept in the art of compression at the time the invention was made. And different techniques of compression taking into consideration such known concept were the likes of Huffman encoding, Lempel-Ziv techniques, or Run-Length or delta encoding.

As for the limitation of a base symbol to which differential encoding operates, Unger teaches compiler information being compressed and using some compression encoding scheme (e.g. *dictionary ... encoding, run-length encoding* - col. 10, line 23 to col. 11, line 44) not only implicitly discloses some form of encoding the textual and numerical symbols under some format required for HTML protocol (e.g. col. 4, lines 25-55) but also discloses compacting repeated characters, e.g. run length of repeated elements, in the sequence of data or encoded text being produced and only storing the portion that is not commonly repeated along with a representation of the repeated portions. Similar to identifying a common repeated strings and appending the rest of the sequences to this common group, another technique by Ainon discloses not the repeated elements but a common ancestor/family group based on a family of word type (see pg. 419-420). It would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the compression by Unger using a base group, or base symbol as claimed, identified as an ancestor or family of word type being encoded via specific integer format as taught by Ainon, because since Unger discloses word coming from a dictionary, using the criteria of identifying common atomic elements based on family or ancestor of word type as taught by Ainon would also be a form of alleviating repeats as mentioned above; but Ainon does not explicitly teach compacting by appending the differential parts to the identified base group.

Analogous to the commonly repeated group by Run-length technique and the dictionary word group type by Ainon, Storer discloses replaced duplicated substrings as suggested by Run-Length with a pointers and appending the non-common portions to the representation of such commonly repeated portions of the sequence (e.g. pg. 30 section 2, to pg. 31, middle right column). In view of the different approach as to represent a common string or repeated sequences as a particular group being represented a special pointer structure or base dictionary group as suggested by Storer and Ainon, respectively, it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the compression as suggested by Unger (i.e. run-length encoding) so that a base group is stored for repeated sequences and append thereto the non-common elements (i.e. differential parts relative to the common parts of the sequences) as suggested by Ainon and Storer, such that the final compressed compiler product would include base symbol for the program symbol along with the differential name having a reduced-size format. The motivation for this would depend on the degree of variations of the source data and when patterns are such that common groups can be identified as by Ainon, then identifying a base group that is repeated among different sequences of encoded strings, and represent such base group with a compacted structure as by Storer while appending to it the differential parts, this differential type of compression would reduce the overall size of the compiled product considerably according the beneficial results gathered from factoring out common parts by differential compacting as mentioned either by Ainon (see pg. 420) and Storer (see section 1 pg. 30).

As per claim 4, Unger discloses encoding an encoded program symbol name in the compiler information (Fig. 8; col. 8. lines 45-52; col. 4, lines 25-55) with an encoded format but does not explicitly specify identifying an encoded program symbol name that is encoded in an extended format encoding. But the fact of encoding by Unger, i.e. applying additional structure or code to hide the original atomic element of sequence of characters, text or token so to encode a source string or stream implicitly discloses extended format of such original source of data or characters otherwise encoding is no longer encoding because it exposes the very format of the original source data.

Unger does not explicitly disclose determining a differential encoding for the encoded symbol name relative to a base symbol, the differential encoding having reduced-size format as compared to the extended format; nor replacing the extended format encoding for the symbol name with the differential encoding. The fact of compacting the base group or common group under some special structure leaving out the other encoded portions has been addressed in claim 1, hence the limitation as to determining a differential encoding relative to a base symbol, replacing said extended format encoding with the differential encoding, such differential encoding being smaller with respect to the original extended format would have been also obvious in light of the rationale as set forth in claim 1 above.

As per claims 5 and 6, Unger discloses determining an encoded program symbol name identifier (e.g. *token* – col. 8, line 54 to col. 9, line 14; *token range* – Fig. 9); and attaching such identifier to the encoding (Fig. 8; *token numbers* -- col .9, lines 39-54; steps *210,212* – Fig. 8 – Note: a string of tokens being identifiable by some integer reads on encoded program symbol name identifier). However, Unger does not disclose if an augmented differential encoding is needed, attaching a encoded program symbol name identifier to the differential encoding; nor does Unger teach that such identifier is a base symbol name. But these limitations have been addressed for obviousness in claim 1 above (with Ainon and Storer's teachings) and are herein rejected for the same rationale therein because identifying a common group, or a base group identified by some structure, to factor out is equivalent to determining whether a augmented differential encoding is needed and attaching the differential encoding to such common base type identifier and attaching thereto the differential encoding as suggested by Ainon and Storer.

As per claim 10, Unger discloses a method for generating uncompressed symbol names being associated (col. 9, lines 5-30; col. 10, lines 23-39) with compiler information, said method comprising:

receiving a source program including one or more program symbols and non-program information (e.g. .g. *vocabulary word, token* – col. 9, lines 5-30; *numeric string, currency symbols* – col. 10, lines 23-39; steps *212-214, 222* – Fig. 8)

identifying a compressed encoded symbol name being associated with compiler information (*token, words, strings* – col. 16, lines 8-17; Fig. 5 – Note: token is compiler information and symbol formatted inside a HTTP protocol is equivalent to being encoded under such HTML format or HTTP protocol);

obtaining information relating to the compressed symbol name (e.g. *dictionaries* – col. 38-55); and

decompressing the compressed encoded symbol name to obtain an encoded symbol name in a uncompressed form (e.g. col. 15, line 60 to col. 16, line 7).

But Unger does not disclose determining whether any program symbol names are in a differential format; and for each such symbol name being in a differential format, extracting a differential program symbol name and a reference to a base symbol for the program symbol; and using such reference to locate the non-differential identifier for the base symbol. In light of the base group teaching by Ainon, and a pointer structure representing the factored out common sequence by Storer, these limitations as to identify a base group, applying a reference to it and use such reference for locating the non-differential representation of the base symbol has been addressed in claim 1 above and the limitation as to decompress the differential program symbol name based on the non-differential name for the base symbol to obtain an encoded program symbol name in a uncompressed form would thereby have been obvious in light of the association compression/decompression as suggested by Unger and in view of the compression techniques as taught by Ainon and Storer as mentioned in claim 1.

As per claim 12, Unger discloses Run-Length encoding which implicitly teaches a combination of small variations and a grouping of repeated elements, hence has suggested a container for grouping the repeated elements in the sequence. Further, Storer teaches a reference representation of a common group serving a basis of the appending of the differential encoding and Ainon teaches a base type group based on dictionary families of words (see claim 1); hence the combination of Unger in view of Ainon/Stored would render this container limitation obvious in light of the rationale as set forth in claim 1.

As per claim 13, Unger discloses a compilation system suitable for compiling source programs, said compilation system comprising:

an enhanced compiler suitable for generation of enhanced compiler products (products *54-62* – Fig. 7), such compiler being operable to compile a source program having at least one program symbol name to produce the enhanced compiler products, such products having a reduced size in comparison with conventional compiler products (e.g. col. 1, line 47 to col. 2, line 39; steps *210, 212, 213, 218* -- Fig. 8); and

at least one enhanced non-compiler component operable to understand and utilize the enhanced compiler products (e.g. *proxy* – col. 14, lines 14-58 – Note: decompressing compressed data is equivalent to parsing and making use of compressed compiler products).

But Unger does not explicitly specify that the source program has at least one compressed encoded program symbol name. But this encoded symbol limitation has been determined as being disclosed in claim 1.

Nor does Unger disclose including one or more differential names corresponding to the program symbol names. But the limitation as to using differential encoding in association with program symbol names has been addressed in claim 1 using the suggested Run-Length encoding by Unger with the teachings by Ainon and Storer.

As per claim 14, Unger discloses using encoding technique to reduce size of the enhanced compiler product (Unger: *Huffman* -- col. 8. lines 45-52; *run-length* – col. 11, lines 38-44); but does not specify that such reduction is up to 40 percent of sizes of conventional compiler products. Ainon, in a analogous method to compress data based on dictionary of words using a family type to form a base group as disclosed in claim 1, discloses a better ratio result up to 40% to conventional methods (Table 2 pg. 420 - *Two-byte-word 2.0* and *Huffman 1.9*). It would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the compressing and encoding by Ainon to the Huffman's technique by Unger because targeting and achieving up to 40% in size reduction would better preserve storage resources as intended in Unger's compression technique.

As per claim 16, this is a computer-readable medium claim corresponding to claim 1 above, including all the limitations therein, hence is rejected herein for the same reasons as set forth therein.

As per claim 19, this is a computer-readable medium claim corresponding to claim 4 above, including all the limitations therein, hence is rejected herein for the same reasons as set forth therein.

As per claim 21, Unger discloses a computer-readable medium including a computer program encoded program symbol names in a uncompressed from and associated with compiler information, said computer medium comprising the corresponding limitations of claim 10 above. Hence the claim is rejected herein for the same reasons as set forth therein.

As per claim 22, this claim includes the base program symbol being a container and corresponds to claim 12; hence is rejected with the rejection as set forth therein.

4. Claims 7-8 are rejected under 35 U.S.C. 103(a) as being unpatentable over Unger et al, USPN: 5,991,173, in view of Ainon, "Storing text using integer codes", 1986, *Proceedings of the 11th coference on Computational linguistics*; and Storer et al., "The Macro Model for Data Compression", 1978, *Proceedings of the 10th Annual ACM symposium on Theory of computing*, as applied to claim 1; and further in view of Porter et al., USPN: 6,163, 811, (hereinafter Porter).

As per claim 7, Unger discloses that the source program to compile is HTML material such as HTML, XML, SGML files (e.g. col. 5, lines 1-12); but does not specify that the source program is a programming language written in C++, Java, Pascal, or Fortran. Porter, in a method to compress application code using tokenized source data and symbol storage and web page for source file (e.g. col. 2, line 21-34) as mentioned above, discloses applying compression to Java program source code (e.g. col. 4, lines 6-24; Figs. 1-3). It would have been obvious for one of ordinary skill in the art at the time the invention was made to use a source program written in Java as taught by Porter and submit it to the compression process used by Unger because Java language programming and its products are well-known for their portability and platform independency as well as support of many browser applications and material, i.e. HTML, XML applications just as suggested in Unger's invention.

As per claim 8, Unger discloses parsing and compressing browser documents but Porter from above discloses compressing of program code using tokenized process analogous to Unger. In view of the rationale in claim 1 using Porter's teachings for addressing the program code symbols parsing, the limitation as to compress an object code file would also have been obvious herein because one ordinary skill in the art would be motivated to combine using the browser compiler/parsing schemes by Unger and enhance those with capabilities to parse program code and compress such code as taught by Porter in order to yield compressed version of such parsed program as intended by Unger, because object code delivery in compressed form would facilitate distribution and storage resources saving.

5. Claims 9, 15, and 20 are rejected under 35 U.S.C. 103(a) as being unpatentable over Unger et al, USPN: 5,991,173 (hereinafter Unger), in view of Ainon, "Storing text using integer codes", 1986, *Proceedings of the 11th coference on Computational linguistics*; and Storer et al., "The Macro Model for Data Compression", 1978, *Proceedings of the 10th Annual ACM symposium on Theory of computing*, as applied to claims 1, 13, and 16 above, and further in view of Porter et al., USPN: 6,163, 811, and (no author) G06F011/28 by Derwent 1998-236084, JP Pub N: JP 10074152A (hereinafter JP-DW-1998).

As per claim 9, Unger (with Ainon and Storer teachings) teaches an enhanced compressed compiler product, but fails to specify including therein an object file, an executable or a debugging information. Porter, in a method to compress application code using tokenized source data and symbol storage and web page for source file (e.g. col. 2, line 21-34) as mentioned above, discloses applying compression to Java program source code (e.g. col. 4, lines 6-24; Figs. 1-3), hence has suggested a program object code as mentioned in claim 8. Further, JP-DW-1998, in a debug system for compressing code as mentioned in claim 8 above, discloses compressing both the debug information and executable code in the deliverable that is to be loaded on the target computer (JP-DW-1998: see front page and ABSTRACT). In view of the rationale in claim 8 to combine Unger teachings with Porter's for providing object file code, it would also be obvious for one of ordinary skill in the art at the time the invention was made to further include executable code and debug information in the compressed product as taught by JP-DW-1998 in order to enhance the utilization of the compressed code delivered as suggested by Porter (in combination with Unger) as to facilitate the debugging and additional memory usage as suggested by JP-DW-1998.

As per claim 15, this claim includes the same limitations as claims 8 and 9 above, hence is rejected herein for the same reasons as set forth therein.

As per claim 20, this claim includes the same limitations as claim 15 above, hence is rejected herein for the same reasons as set forth therein.

Response to Arguments

6. Applicant's arguments filed 6/1/2004 have been fully considered. Most arguments are becoming moot in regard to the new grounds of rejection. With respect to some arguments following are the Examiner's observation and response thereto.

(A) As per claims 1 and 16, Applicants have submitted that Unger discloses words of a dictionary with the symbols which are but mere characters in a text file and thus does not disclose encoding program symbol names without changing the non-program symbol information (Appl. Rmrks, pg. 7, last para, pg. 8, top 3 para). As interpreted, the claimed features do not enforce how a particular or specific format under which a program symbol name is encoded for subsequent compression while excluding the rest of a document symbols. The rejection has pointed out that Unger uses some scheme to compress textual content from markup files, and by submitting textual data under some markup process, some level of special encoding is implicitly disclosed. It is well recognized that rendering a HTML, SGML or XML document is also a form of encoding being applied to textual tokens or symbols to conform those respective standards. Thus, creating specially formatted documents like markup templates or source code lines as in a browser page or markup files already encompasses a certain level of encoding via some formatting scheme or algorithm according to the language format, type, inter-relationship and grammar because encoding is interpreted as working on the original set of data so to produce a modified set of data using some reformatting scheme or rule for different purposes. For instance, source code is usually encoded into format understandable by low-level execution machine or runtime engine. And this is exemplified in markup code being re-formatted into a special arrangement scheme in order to be interpreted by browser execution engine. Thus, the encoding from regular text into markup as by Unger fits this description. The claim does not specify how the limitation recited as 'encoding a program symbol name to produce an encoded program symbol name' clearly distinguishes over the process of compressing or encoding using markup formatting by Unger. Further, the recited 'without changing the non-program symbol information' does not spell out that only some form of symbols are encoded while leaving out some other forms of textual content. The rejection as set forth also points out which part of the compiler product being encoded is left out and which part is encoded in the compressing scheme.

(B) As per the arguments concerning Porter with respect to claim 1(Appl. Rmrks, pg. 8, bottom para), these are moot in light of the now current rejection.

(C) As for the rest of the arguments (Appl. Rmrks, pg. 9), these are based on the amended claims and are also moot in view of the new grounds of rejection.

Conclusion

7. In view of the amendments which necessitated new grounds of rejections, **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (703)305-7207. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (703)305-9662.

Any response to this action should be mailed to:

Commissioner of Patents and Trademarks

Washington, D.C. 20231

or faxed to:

(703) 872-9306 (for formal communications intended for entry)

or: (703) 746-8734 (for informal or draft communications, please label "PROPOSED" or "DRAFT" – please consult Examiner before use)

Hand-delivered responses should be brought to Crystal Park II, 2121 Crystal Drive, Arlington. VA. , 22202. 4th Floor(Receptionist).

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-3900.

VAT
August 30, 2004

[illegible]

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100